Razvoj interaktivne Escape room igre u Unityu za virtualnu stvarnost

Ružić, Leo

VELEUČILIŠTE U BJELOVARU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

RAZVOJ INTERAKTIVNE ESCAPE ROOM IGRE U UNITYU ZA VIRTUALNU STVARNOST

Završni rad br. 06/RAČ/2024

Leo Ružić

Bjelovar, rujan 2024.

obrazac ZR - 001

Veleučilište u Bjelovaru

Trg E. Kvaternika 4, Bjelovar

1. DEFINIRANJE TEME ZAVRŠNOG RADA I POVJERENSTVA

Student: **Leo Ružić** JMBAG: **0314022956**

Naslov rada (tema): **Razvoj interaktivne Escape room igre u Unityu za virtualnu stvarnost**

Područje: **Tehničke znanosti** Polje: **Računarstvo**

Grana: **Programsko inženjerstvo**

Mentor: **Krešimir Markota, mag. ing. comp.** zvanje: **predavač**

Članovi Povjerenstva za ocjenjivanje i obranu završnog rada:

1. **Tomislav Adamović, mag. ing. el., predsjednik**
2. **Krešimir Markota, mag. ing. comp., mentor**
3. **Krunoslav Husak, dipl. ing. rač., član**

2. ZADATAK ZAVRŠNOG RADA BROJ: 06/RAČ/2024

U sklopu završnog rada potrebno je:

1. Analizirati i opisati potrebne alate koji će se koristiti prilikom izrade Escape room video igre za virtualnu stvarnost
2. Osmisliti zaplet, osnovne ciljeve i strukturu igre
3. Izraditi 3D okruženje za Escape room igru pomoću integriranih Unity modela
4. Izraditi programsku podršku za integraciju virtualne stvarnosti u Unity okruženju pomoću XR Unity biblioteke
5. Izraditi osnovnu logiku upravljanja mehanikama igre i upravljanja scenama unutar Unity okruženja

Datum: 12. srpnja 2024. godine Mentor: **Krešimir Markota, mag. ing. comp.**

Sadržaj

1. UVOD

U ovom radu bit će predstavljena izrada VR igre, koje su se ideje htjele prikazati i kako su se te ideje uspjele izvesti.

Igra je izrađena u stilu *escape rooma*, gdje je igračev cilj bijeg iz prostorije u kojoj se trenutno nalazi koristeći objekte u prostoriji. Elementi igre su izrađeni u Unity programu i okruženju za razvijanje videoigara, zajedno s Unity XR paketom i korištenjem Visual Studia. Unity je izabran za izradu videoigre zbog dostupne studentske licence, količine primjera i dokumentacije te integriranosti s Visual Studiom. Unity XR paket je obvezan za korištenje jer bez njega nije moguće izraditi VR igru u Unity programu. Igra je testirana na Oculus VR *headsetu* zajedno s upravljačima i opremom s kojima *headset* dolazi.

U ovom radu će se prikazati razmišljanje tijekom izrade VR igre te kako su osmišljeni i izrađeni njezini sastavni dijelovi. Svi elementi igre bit će opisani redoslijedom kojim se pojavljuju u virtualnim prostorijama VR igre. Opisan će biti koncept kloniranih ruku i kako se taj koncept može proširiti.

Sadržaj rada je podijeljen na četiri poglavlja. Prvo poglavlje pojašnjava što je to *escape room* i princip izrade igre u tome stilu. Drugo poglavlje ukratko opisuje korištene tehnologije i za koje su dijelove igre korištene. Treće poglavlje opisuje sve elemente igre, koja je njihova svrha i koje su njihove mogućnosti. Četvrto i zadnje poglavlje opisuje mehanike koje igrač koristi za rješavanje zagonetki postavljenih u igri, zajedno s nekoliko prostorija iz same igre kao primjere.

2. ŠTO JE *ESCAPE ROOM*?

Igračev cilj u bilo kojem *escape roomu* je pronaći izlaz van. Izazov se nalazi u zagonetkama i misterijima koje igrač treba riješiti kako bi izišao iz prostorije ili više prostorija i pobijedio. *Escape room-ovi* su najčešće iskustvo za dvoje ili više osoba koje zajedno surađuju. Više o njihovom sadržaju može se pronaći na [1].

Escape room igre vuku korijene iz karnevalskih atrakcija kao uklete kuće ili potrage za objektima na određenim kontroliranim lokacijama. Prvi *escape room,* u stilu kakav je prikazan u ovoj igri, predstavljen je 2003. godine na GenCon Indy u Indianapolisu. Nakon prvog predstavljanja *escape room-ovi* su nastavili evoluirati s vremenom, prelazeći sa zagonetki s papirom i olovkom, na zagonetke koje uključuju fiziku, matematiku i druge fizičke objekte kao lokote, *puzzle* ili šah. Dodatne informacije o povijesti *escape rooma* nalaze se na [1].

Najveći izazov tijekom izrade bilo kakvog *escape rooma* je osigurati sigurnost igrača. Svi elementi zagonetki moraju biti sigurni za ljudsku interakciju i bez previše oštrih rubova. Također treba osigurati protok zraka u prostoriju tijekom igre i moći motriti stanje igre u slučaju nezgoda. U slučaju većih nesreća i opasnosti potrebno je osigurati rutu za bijeg van iz prostorije ili *panic button* koji igrači mogu stisnuti ako je potrebno. Dodatni primjeri poboljšanja sigurnosti u *escape roomu* mogu se pročitati na [2].

U igrama, naročito VR igrama, koncept *escape rooma* može se dodatno proširiti sa zagonetkama koje ne moraju poštivati zakone fizike u stvarnom svijetu. Primjer VR *escape room* igre koja se koristi takvim mogućnostima jest The Room VR: A Dark Matter. Ta se igra koristi magijom i nemogućom tehnologijom u svojim zagonetkama. Sadržaj cijele igre može se vidjeti na [3]. Također, lakše je osigurati sigurnost igrača zato što igrač ne treba fizičku interakciju s objektima igre. Ali, svakako je potrebno osigurati da se igrač ne prestraši previše ili ne dobije epileptični napadaj dodavanjem upozorenja na sadržaj igre tijekom pokretanja.

3. KORIŠTENE TEHNOLOGIJE I PROGRAMI

Izrada VR igre zahtijevala je tehnologije s pomoću kojih se može izraditi igra i prilagoditi je za VR *headset*. Korištene su četiri tehnologije. To su Unity, Visual Studio, XR Interaction Tool kit i Git. Uz svaku je tehnologiju opisano što je izrađeno pomoću nje. Sve korištene tehnologije javno su dostupne i besplatne za korištenje.

3.1 Unity

Program u kojem su izrađeni svi elementi i prostorije u videoigri je Unity, program i okruženje za razvijanje videoigara. Korištena verzija programa je 2022.3.17f1. Unity sadrži alate za izradu objekata iz jednostavnih oblika poput kocki, sfera i ostalih geometrijskih tijela i likova. Na tim objektima moguće je definirati vrijednosti veličine i rotacije zajedno sa svojstvima objekta. Na objekte se također mogu dodati skripte koje se pišu unutar Visual Studia i daju objektima dodatne mogućnosti. Sam program je detaljnije opisan u Unity dokumentaciji na [4].

3.2 Visual Studio

Cijeli kod koji se koristi u videoigri napisan je u Microsoft Visual Studiu. Korištena inačica programa je Visual Studio Community 2022, verzija 17.8.4. Unity paket za kod omogućava Unityu čitati i razumjeti kod kako bi elementi igre radili kako treba. Visual Studio služi za pisanje i izvođenje koda u C# jeziku pomoću Unity paketa. Sadrži funkcije i tipove podataka unikatne za Unity program koje on razumije i izvodi. U skriptama se definiraju mogućnosti objekata koji ih sadrže. Program i korištenje Unity paketa dodatno je opisano u dokumentaciji na [5].

3.3 XR Interaction Toolkit

Paket izrađen specifično za pomaganje korisnicima tijekom izrade VR igara u Unityu je XR Interaction Toolkit. Korištena verzija paketa je 2.5.2. XR paket sadrži sve osnovne skripte, elemente za izradu i upravljanje VR igrom te međukod da Unity razumije unos iz VR headseta. Skripte koje se nalaze u paketu služe kao unaprijed izrađene mogućnosti za objekte i igrača. Korištenje i instalacija paketa dodatno su opisani u dokumentaciji na [6].

3.4 Git

Program korišten za spremanje i kontrolu promjena nad projektom je Git. Korištena verzija je 2.41.0.windows.1. Git se nakon instalacije pokreće preko *command linea* i stvara se repozitorij u datoteci gdje se koristi za kontrolu verzije. Pomoću naredbi dostupnih na [7] možemo upravljati resursima koji se nalaze u datoteci i osigurati dostupnost projekta na više uređaja preko usluga kao što su GitHub ili GitLab. U ovom projektu korišten je privatni repozitorij na GitLabu.

4. ELEMENTI I OBJEKTI IGRE

Važan sastav igre su njezini elementi i objekti koje igrač koristi za rješavanje zagonetki. Svi elementi podupiru korištenje skripte SetColor, koja omogućava mijenjanje boje elemenata po želji.

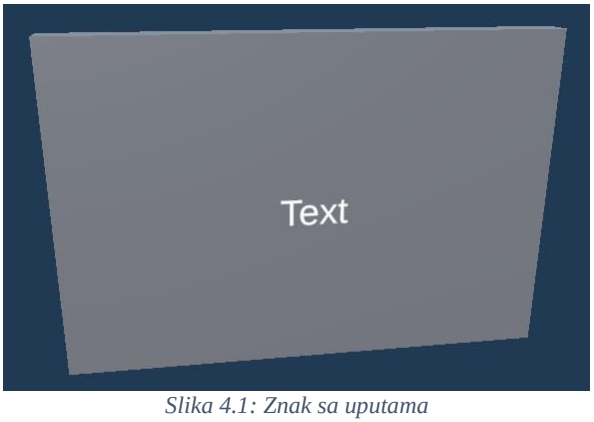
SetColor skripta, prikazana na kodu 4.1, postavlja željenu boju, definiranu RGB vrijednostima, na sve elemente od kojih se sastoji objekt. Skripta to radi tako da pretraži objekt i promijeni boju na objektu i djeci objekta. Sama skripta je prekopirana iz razgovora koji se nalazi na [8].

Programski kod 4.1: SetColor – C#

```
public class SetColor : MonoBehaviour
{
   public Color c;
   void Start()
   {
      foreach (Renderer r in GetComponentsInChildren<Renderer>())
      {
         r.material.color = c;
      }
   }
}
```

4.1 Znak s uputama

Element koji se nalazi u svakoj prostoriji i služi kako bi pomogao igraču upoznati se s igrom ili usmjerio ga u pravom smjeru je znak s uputama, prikazan na slici 4.1. Sastoji se od velikog kvadrata s tekstnim sadržajem. Tekst koji se nalazi na znaku pokazuje upute upravljanja svijetom ili ponekad humoristični sadržaj.

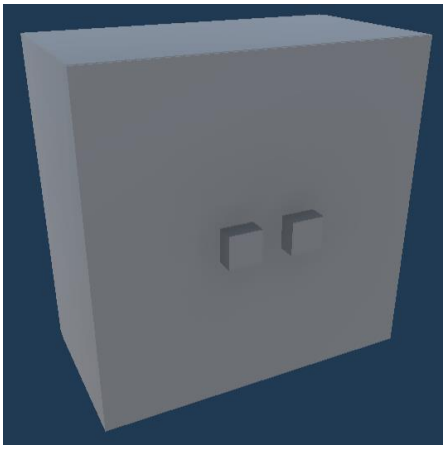


Slika 4.1: Znak sa uputama

4.2 Ormarić

Igrač tijekom igre treba pronaći druge elemente koji će mu pomoći riješiti zagonetku. Ti se elementi često nalaze unutar elemenata za pohranu. Prvi od takvih elemenata je

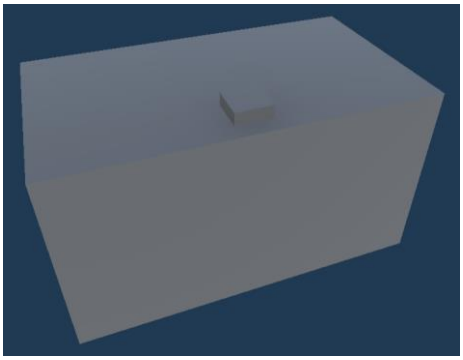
ormarić, prikazan na slici 4.2. Sastoji se od pet kvadratnih stranica i dvojih kvadratnih vrata s ručkama. Igrač može primiti ručke kako bi otvorio vrata i uzeo objekte koji se nalaze u ormariću. U ormariću se često nalaze ključevi ili drugi objekti potrebni za rješavanje trenutne zagonetke.



Slika 4.2: Ormarić

4.3 Kovčeg

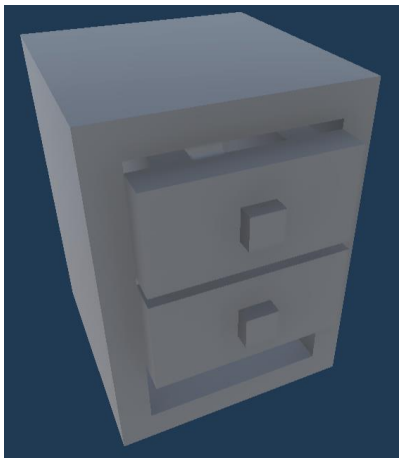
Drugi od elemenata za pohranu je kovčeg, prikazan na slici 4.3. Sastoji se od pet stranica i poklopca s ručkom. Igrač može primiti ručku kako bi otvorio poklopac i uzeo stvari koje se nalaze u kovčegu. U kovčegu se često nalaze ključevi ili drugi objekti potrebni za rješavanje trenutačne zagonetke.



Slika 4.3: Kovčeg

4.4 Ladice

Posljednji element za pohranu su ladice, prikazane na slici 4.4. Sastoje se od pet kvadratnih stranica i dvije male ladice. Male se ladice sastoje od pet kvadratnih stranica i ručke. Igrač može primiti ladice za ručku kako bi ih izvukao i uzeo što se nalazi u njima. U ladicama se često nalazi objekt potreban za rješavanje zagonetke, primjerice ključ ili papir s ispravnom kombinacijom. Primjer izrade ladica nalazi se na [9] gdje je objašnjeno kako osigurati da se ladice mogu ispravno izvlačiti.



Slika 4.4: Ladica

4.5 Vrata

Element koji služi kao prijelaz iz prostorije u prostoriju ili cilj koji treba otvoriti kako bi igrač nastavio dalje su vrata, prikazana na slici 4.5. Sastoje se od okvira sastavljenog od triju kvadrata i samih kvadratnih vrata s kvakom. Vrata se otvaraju tako da ih igrač pogurne ili primi za kvaku. Primjer izrade vrata nalazi se na [9,10] gdje je objašnjeno kako osigurati ispravnu rotaciju vrata prema okviru.

Slika 4.5: Vrata

4.6 Ljestve

Alternativni prijelaz u prostorije umjesto vrata su ljestve, prikazane na slici 4.6. One se koriste kada se sljedeća prostorija nalazi iznad prethodne, ili kao pomoć igraču da može doći do inače nedostupnih visina. Sastoje se od vertikalnih prečki i horizontalnih ručki. Igrač se može primiti za ručke kako bi se povukao prema gore i popeo pomoću ljestvi.

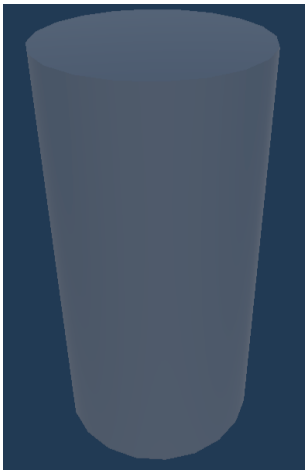
Slika 4.6: Ljestve

4.7 Teleporteri

Svi elementi koji prenose određene objekte na određenu lokaciju su teleporteri. Njihova svrha je ubrzati ili olakšati povratak igrača ili važnog objekta na važnu lokaciju u svrhu rješavanja zagonetke. Teleporteri se razlikuju samo po obliku i po tome koje objekte mogu teleportirati.

4.7.1 Teleporter igrača

Prva vrsta teleportera je teleporter igrača, koji služi kako bi prebacio igrača na određenu poziciju, prikazan na slici 4.7. Sastoji se samo od cilindričnog *collidera*. Teleporter sadrži skriptu PlayerTeleport kako bi ispravno teleportirao igrača.



Slika 4.7: Teleporter igrača

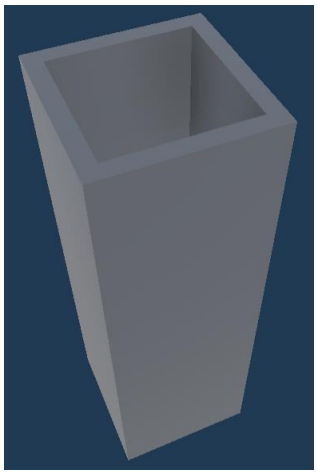
PlayerTeleport skripta, prikazana na kodu 4.2, traži unos vrijednosti pozicije na koju će teleporter teleportirati igrača. Kada igrač dođe u kontakt s teleporterom, kontakt se detektira i pronalaze se igračeve ruke nakon same teleportacije igrača. Ruke se spremaju u listu i pojedinačno teleportiraju na poziciju igrača.

Programski kod 4.2: PlayerTeleport – C#

```
public class PlayerTeleport : MonoBehaviour
{
   public float x, y, z;
   private HandPresence[] hands;
   private void OnTriggerEnter(Collider otherobject)
   {
      if (otherobject.CompareTag("Player"))
      {
         otherobject.transform.position = new Vector3(x, y, z);
         hands = FindObjectsOfType<HandPresence>();
         foreach (HandPresence hand in hands)
         {
            hand.transform.position = new Vector3(x, y, z);
         }
      }
   }
}
```

4.7.2 Teleporter ključa

Druga vrsta teleportera, koji služi kako bi prebacio ključ na određenu poziciju, je teleporter ključa, prikazan na slici 4.8. Sastoji se od četvrtaste cijevi s pet kvadratnih stranica i samog teleportera sastavljenog samo od *collidera*. Teleporter sadrži skriptu KeyTeleport kako bi ispravno teleportirao ključ.



Slika 4.8: Teleporter ključa

KeyTeleport skripta, prikazana na kodu 4.3, traži unos vrijednosti pozicije na koju će teleporter teleportirati ključ. Kada ključ dođe u kontakt s teleporterom na dnu cijevi, on se teleportira na unesenu poziciju.

Programski kod 4.3: KeyTeleport – C#

```
public class KeyTeleport : MonoBehaviour
{
   public float x, y, z;
   private void OnTriggerEnter(Collider otherobject)
   {
      if (otherobject.CompareTag("Key"))
      {
         otherobject.transform.parent.position = new Vector3(x, y, z);
      }
   }
}
```

4.7.3 Teleporter ruku

Posljednja vrsta teleportera, koja služi za prebacivanje kloniranih ruku na određenu poziciju, je teleporter ruku, prikazan na slici 4.9. Najčešće se koristi kako bi igrač mogao riješiti zagonetku. Sastoji se samo od sferičnog *collidera*. Teleporter sadrži skriptu HandTeleport kako bi ispravno teleportirao klonirane ruke.

Slika 4.9: Teleporter ruku

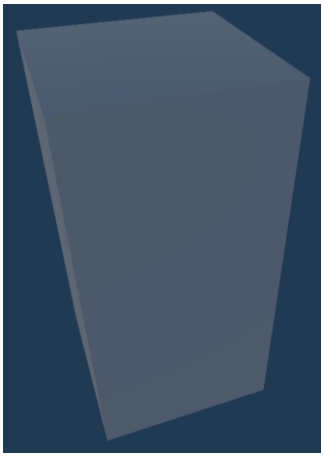
HandTeleport skripta, prikazana na kodu 4.4, traži unos vrijednosti pozicije na koju će teleporter teleportirati klonirane ruke. Kada klonirana ruka bilo kojega tipa dođe u kontakt s teleporterom, ona se teleportira na unesenu poziciju.

Programski kod 4.4: HandTeleport – C#

```
public class HandTeleport : MonoBehaviour
{
   public float x, y, z;
   private void OnTriggerStay(Collider otherobject)
   {
      if (otherobject.CompareTag("Teleport Hand") || otherobject.CompareTag("Freeze Hand")
|| otherobject.CompareTag("Climb Hand"))
      {
         otherobject.transform.position = new Vector3(x, y, z);
      }
   }
}
```

4.8 Zona za promjenu ruke

Igrač treba imati mogućnost mijenjanja trenutačno dostupnih kloniranih ruku. Element koji tu mogućnost omogućava je zona za promjene ruke, prikazana na slici 4.10. Sastoji se samo od kvadratnog *collidera*. Zona sadrži skriptu HandToggle koja upravlja promjenom igračevih ruku.



Slika 4.10: Zona za promjenu ruke

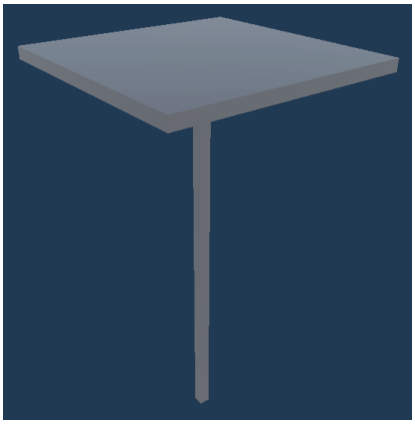
HandToggle skripta, prikazana na kodu 4.5, provjerava nalazi li se igrač u njoj. Ako se nalazi, postavlja igračeve trenutačne klonirane ruke na ruke određene u skripti, vraćajući ruke koje igrač ne koristi na početne pozicije. U skripti se definira koje ruke želimo da igrač koristi, a koje ne.

Programski kod 4.5: HandToggle – C#

```
public class HandToggle : MonoBehaviour
{
   public GameObject LeftHand, RightHand, OtherLeftHand, OtherRightHand, LeftHandToSetTo,
RightHandToSetTo;
   public CloneHandController controllerleft, controllerright;

   private void OnTriggerEnter(Collider otherobject)
   {
      if (otherobject.CompareTag("Player"))
      {
         controllerleft.cloneHand = LeftHandToSetTo;
         controllerright.cloneHand = RightHandToSetTo;
         LeftHand.transform.position = new Vector3(0, -1, 0);
         RightHand.transform.position = new Vector3(0, -1, 0);
         LeftHand.transform.rotation = Quaternion.Euler(0, 0, 90);
         RightHand.transform.rotation = Quaternion.Euler(0, 0, -90);
         OtherLeftHand.transform.position = new Vector3(0, -1, 0);
         OtherRightHand.transform.position = new Vector3(0, -1, 0);
         OtherLeftHand.transform.rotation = Quaternion.Euler(0, 0, 90);
         OtherRightHand.transform.rotation = Quaternion.Euler(0, 0, -90);
      }
   }
}
```

4.9 Pokretna platforma

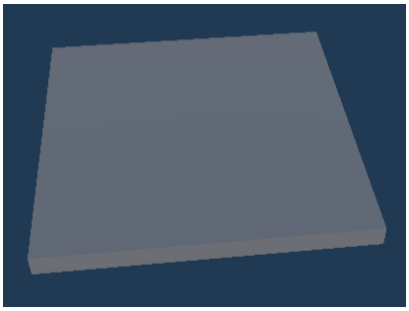
Igrač tijekom igre povremeno mora preći s jedne na drugu udaljenu platformu. Jedan od elemenata, koji služi kao pomoć igraču kako bi prešao razmake između tih platformi, je pokretna platforma, prikazana na slici 4.11. Sastoji se od tankog kvadratnog štapa i široke kvadratne platforme. Njezina visina nije prilagodljiva i često služi samo za određene razmake.



Slika 4.11: Pokretna platforma

4.10 Prenosiva platforma

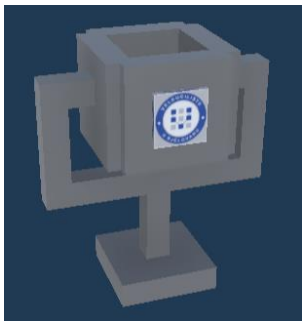
Drugi element koji služi kao pomoć igraču kao bi prešao razmake između razmaknutih platformi je prenosiva platforma, prikazana na slici 4.12. Sastoji se samo od široke kvadratne platforme. Za razliku od pokretne platforme ima prilagodljivu visinu, ovisno o tome gdje je zamrznuta.



Slika 4.12: Prenosiva platforma

4.11 Trofej

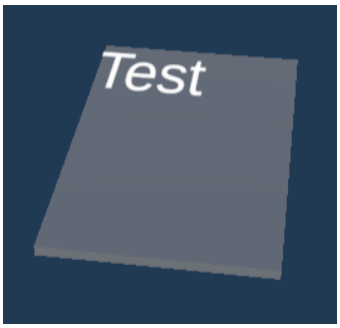
Element koji sadrži VUB-ov logo i služi kao nagrada za završavanje igre je trofej, prikazan na slici 4.13. Sastoji se od kvadratne baze, kvadratnog štapa za vrat, tri kvadratna štapa za ručke, pet kvadrata koji su formirani u obliku kupa i jedan kvadrat s logom na samom kupu. Trofej se nalazi samo u završnoj prostoriji igre.



Slika 4.13: Trofej

4.12 Papir

Igrač treba moći saznati ispravnu kombinaciju za različite kombinacijske zagonetke. Element koji služi kako bi pomogao igraču s kombinacijski zagonetkama je papir, prikazan na slici 4.14. Sastoji se samo od širokog i tankog kvadrata s tekstom. Najčešće sadrži točne kombinacije koje igrač mora unesti kako bi napredovao u prostoriji.



Slika 4.14: Papir

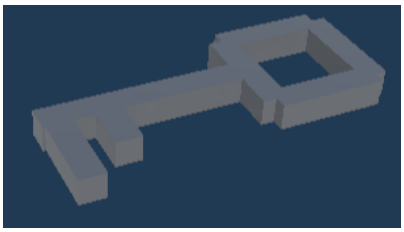
4.13 Lopta

Elementi koje igrač često pronalazi unutar elemenata za pohranu i često se koriste za otključavanje lokota se nazivaju ključni elementi. Jedan od ključnih elemenata je lopta, prikazana na slici 4.15. Sastoji se samo od sfere. Najčešće služi za otključavanje lokota zajedno s detektorom objekta.

Slika 4.15: Lopta

4.14 Ključ

Alternativni ključni element umjesto lopte je ključ, prikazan na slici 4.16. Sastoji se od sedam kvadrata posloženi da izgledaju kao ključ. Najčešće služi za otključavanje lokota tako da se sam ključ približi lokotu.



Slika 4.16: Ključ

4.15 Lokot

Element koji služi za zaključavanje ili blokiranje različitih važnih elemenata je lokot, prikazan na slici 4.17. Igračev je zadatak otključati lokote kako bi nastavio dalje. Sastoji se od velikog kvadrata, odnosno donjeg dijela lokota i tri manja kvadrata koji čine gornji dio lokota. Sami se lokoti mogu otključati na različite načine, najčešće s ključem ili rješavanjem zagonetke. Lokot može sadržavati tri različite skripte koje definiraju kako se lokot otključava. Te skripte su OpenCombinationDoor, OpenDetectorDoor i OpenKeyDoor.

Slika 4.17: Lokot

OpenCombinationDoor skripta, prikazana na kodu 4.6, provjerava je li unesena ispravna kombinacija na kombinacijskom satu, ladicama ili gumbima koji se promatraju, otključavajući lokot. OpenDetectorDoor skripta, prikazana na kodu 4.7, provjerava nalazi li se ispravan objekt u detektoru objekta, otključavajući lokot. OpenKeyDoor skripta, prikazana na kodu 4.8, provjerava nalazi li se u *collideru* lokota ispravan objekt koji ga otključava, najčešće ključ. Sve skripte dijele mogućnost sprečavanja interakcije sa zaključanim objektima. Sprečavaju njihovo zamrzavanje pomoću ruku za zamrzavanje te

mogućnost primanja objekta i pretvaraju objekt u kinematičko stanje. Nakon otključavanja objekti dobivaju te mogućnosti i prestaju biti kinematički.

Programski kod 4.6: OpenCombinationDoor – C#

```
public class OpenCombinationDoor : MonoBehaviour
{
    …

    void Update()
    {
        if (open.correct)
        {
            rb.isKinematic = false;
            interactable.enabled = true;
            freeze.enabled = true;
            Destroy(gameObject);
        } else
        {
            freeze.enabled = false;
            interactable.enabled = false;
            rb.isKinematic = true;
        }
    }
}
```

Programski kod 4.7: OpenDetectorDoor – C#

```
public class OpenDetectorDoor : MonoBehaviour
{
    …

    void Update()
    {
        if (open.itemDetected)
        {
            rb.isKinematic = false;
            interactable.enabled = true;
            freeze.enabled = true;
            Destroy(gameObject);
        } else
        {
            freeze.enabled = false;
            interactable.enabled = false;
            rb.isKinematic = true;
        }
    }
}
```

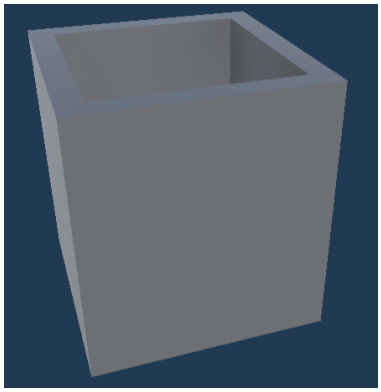
Programski kod 4.8: OpenKeyDoor – C#

```
public class OpenKeyDoor : MonoBehaviour
{
    …

    private void OnTriggerEnter(Collider otherObject)
    {
        if (otherObject.gameObject == key || otherObject.transform.parent.gameObject == key)
        {
```

```
            rb.isKinematic = false;
            interactable.enabled = true;
            freeze.enabled = true;
            Destroy(key);
            Destroy(gameObject);
        }
    }

    ...
}
```

4.16 Detektor objekta

Jedan od alternativnih načina otključavanja lokota je detektor objekta, prikazan na slici 4.18. Sastoji se od pet kvadrata koji čine košaru i *collidera* unutar košare. On gleda objekt koji se ubaci u njega pa ako je detektiran ispravan objekt, otključava lokot. Detektor sadrži skriptu ItemDetectorValue.



Slika 4.18: Detektor objekta

ItemDetectorValue skripta, prikazana na kodu 4.9, detektira objekt unutar *collidera* pa ako je objekt ispravan, tu vrijednost šalje dalje na lokot s OpenDetectorDoor skriptom koja otključava lokot.

Programski kod 4.9: ItemDetectorValue – C#

```
public class ItemDetectorValue : MonoBehaviour
{
    public bool itemDetected;
    public GameObject itemToLookFor;

    private void OnTriggerEnter(Collider otherObject)
    {
        if (otherObject.gameObject == itemToLookFor || otherObject.transform.parent.gameObject == itemToLookFor)
        {
            itemDetected = true;
        }
    }

    private void OnTriggerExit(Collider otherObject)
    {
        if (otherObject.gameObject == itemToLookFor || otherObject.transform.parent.gameObject == itemToLookFor)
        {
            itemDetected = false;
```

```
    }
  }

  void Start()
  {
    itemDetected = false;
  }
}
```

4.17 Kombinacijski elementi

Drugi alternativni način otključavanja lokota su kombinacijski elementi. Svi elementi koji zahtijevaju unos određene kombinacije za rješavanje zagonetke su kombinacijski elementi. Svi kombinacijski elementi sadrže skripte SwitchValue i CombinationValue. SwitchValue skripta, prikazana na kodu 4.10, stavlja se na zasebne gumbe kombinacijskih elemenata. Skripta prati nalazi li se ili se ne nalazi u *collideru* određeni element. Ako je vrijednost ispravna, šalje se na skriptu CombinationValue. CombinationValue skripta, prikazana na kodu 4.11, provjerava jesu li sve dobivene vrijednosti iz SwitchValue skripti ispravne. Ako jesu, vrijednost se šalje na lokot s OpenCombinationDoor skriptom koja otključava lokot. Također, mijenja boju elementa koji govori igraču je li unio ispravnu kombinaciju iz crvene u zelenu.

Programski kod 4.10: SwitchValue – C#

```
public class SwitchValue : MonoBehaviour
{
  public bool activeOnStay;
  public bool buttonActive;

  private void OnTriggerStay()
  {
    if (activeOnStay)
    {
      buttonActive = true;
    }
    else buttonActive = false;
  }

  private void OnTriggerExit()
  {
    if (activeOnStay)
    {
      buttonActive = false;
    }
    else buttonActive = true;
  }

  void Start()
  {
    buttonActive = false;
  }
}
```

Programski kod 4.11: CombinationValue – C#

```
public class CombinationValue : MonoBehaviour
{
    public SwitchValue pull1, pull2, pull3, pull4;
    public Renderer r;
    public bool correct;

    void Start()
    {
        correct = false;
    }

    void Update()
    {
        if (pull1.buttonActive && pull2.buttonActive && pull3.buttonActive && pull4.buttonActive)
        {
            r.material.color = Color.green;
            correct = true;
        }
        else {
            r.material.color = Color.red;
            correct = false;
        }
    }
}
```

4.17.1 Kombinacijski sat

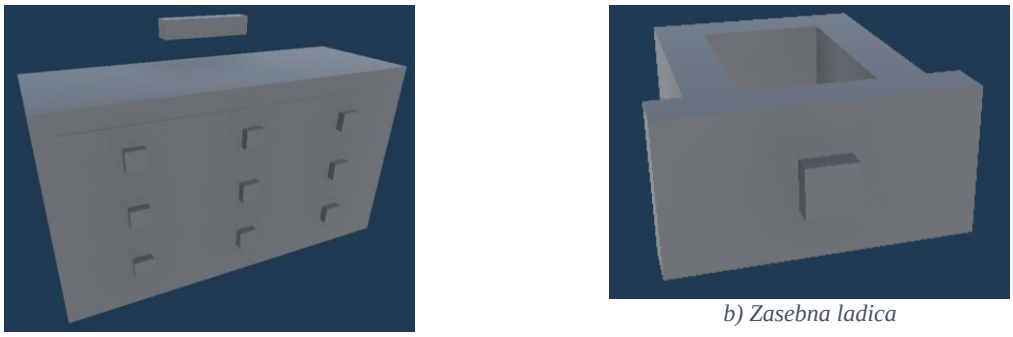
Jedan od kombinacijskih elemenata je kombinacijski sat, prikazan na slici 4.19. Sastoji se od kruga, na kojemu se nalaze zasebni kvadrati koji predstavljaju brojeve, i od tankih kvadratnih štapova koji predstavljaju kazaljke. On traži da se njegove ruke postave u određene pozicije kako bi se otključao lokot ili slično.

Slika 4.19: Kombinacijski sat

4.17.2 Kombinacijske ladice

Drugi kombinacijski elemenat su kombinacijske ladice, prikazane na slikama 4.20. Sastoje se od pet kvadrata koji između sebe imaju devet ladica. Zasebne se ladice sastoje od pet kvadrata s ručkom, malenog kvadrata koji zaustavlja ladicu i *collidera* koji

provjerava stanje zasebne ladice. One traže da se njezine ladice postave u određene pozicije kako bi se otključao lokot ili slično.



a) Sve zajedno

b) Zasebna ladica

Slika 4.20: Kombinacijske ladice

4.17.3 **Kombinacijski gumbi**

Zadnji kombinacijski element su kombinacijski gumbi, prikazani na slikama 4.21. Sastoje se od devet kvadrata koji služe za razdvajanje zasebnih gumba. Zasebni se gumbi sastoje od velikog kvadrata, malenog kvadrata koji zaustavlja veći i *collidera* koji provjerava stanje gumba. Oni traže da se njihovi gumbi pritisnu kako bi se otključao lokot ili slično.

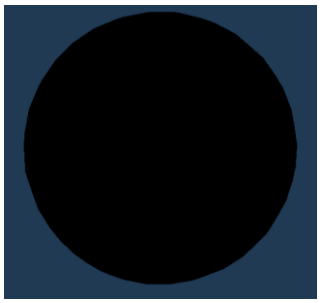
a) Sve zajedno

b) Zasebni gumb

Slika 4.21: Kombinacijski gumbi

4.18 Zacrnjivač pogleda

Element koji služi kako bi igraču zacrnio pogled ako mu glava završi preblizu ili unutar zida je zacrnjivač pogleda, prikazan na slici 4.22. Sastoji se samo od crne sfere. Igračeva glava sadrži skriptu ViewBlockerController koja uključuje vidljivost zacrnjivača pogleda.



Slika 4.22: Zacrnjivač pogleda

ViewBlockerController skripta, prikazana na kodu 4.12, uključuje vidljivost zacrnjivača pogleda kada god je igračev *collider* glave preblizu zidu ili drugim objektima. Primjer izrade zacrnjivača pogleda vidi se na [11,12], gdje je opisana izrada zacrnjivača s postepenim crnilom, za razliku od trenutnog crnila kako je napravljeno ovdje. Opisano je i kako izgurati igrača van zida.

Programski kod 4.12: ViewBlockerController – C#

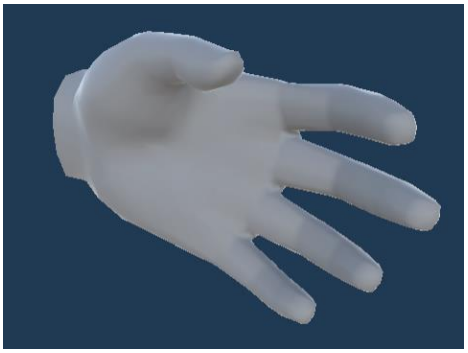
```
public class ViewBlockerController : MonoBehaviour
{
   public GameObject blocker;
   private void OnTriggerEnter(Collider otherobject)
   {
      if (otherobject.CompareTag("Wall"))
      {
         blocker.GetComponent<Renderer>().enabled = true;
      }
   }
   private void OnTriggerExit(Collider otherobject)
   {
      if (otherobject.CompareTag("Wall"))
      {
         blocker.GetComponent<Renderer>().enabled = false;
      }
   }
}
```

5. MEHANIKE IGRE

5.1 Igračeve ruke

Elementi koji služe za igračevu interakciju sa svijetom igre su igračeve ruke, prikazane na slikama 5.1. One su zasebni objekti koje prate nevidljive kontrolere. To se omogućava pomoću skripti HandPresencePhysics i HandPresence. Ruke također sadrže skriptu CloneHandController. Igračeve ruke i njihove mogućnosti izrađene su pomoću baze iz javno dostupnog paketa u opisu videa na [13], gdje se također opisuje kako se ruke mogu izraditi samostalno.

a) Desna ruka



b) Lijeva ruka

Slika 5.1: Igračeve ruke

HandPresencePhysics skripta, prikazana na kodu 5.1, brine se za to da igračeve ruke prate nevidljive kontrolere u kojem god se smjeru kretale. To se ostvaruje tako da skripta prati poziciju i rotaciju kontrolera i stalno mijenja poziciju objekata ruku ovisno o njima. HandPresence skripta, prikazana na kodu 5.2, brine se za to da igračeve ruke imaju animacije za primanje objekata. Skripta detektira pritisak gumba i pokreće animaciju ovisno o tome. CloneHandController skripta, prikazana na kodu 5.3, brine se za to da se klonirane ruke stvore na poziciji objekata ruku, kako bi se spriječilo stvaranje ruku izvan zidova igre. Klonirane se ruke stvaraju pritiskom na Y/B gumb kontrolera.

Programski kod 5.1: HandPresencePhysics – C#

```
public class HandPresencePhysics : MonoBehaviour
{
    ...

    void FixedUpdate()
    {
        rb.velocity = (target.position - transform.position) / Time.fixedDeltaTime;
        Quaternion rotationDifference = target.rotation * Quaternion.Inverse(transform.rotation);
        rotationDifference.ToAngleAxis(out float angleInDegree, out Vector3 rotationAxis);
        Vector3 rotationDifferenceInDegree = angleInDegree * rotationAxis;
        rb.angularVelocity = (rotationDifferenceInDegree * Mathf.Deg2Rad / Time.fixedDeltaTime);
    }
}
```

Programski kod 5.2: HandPresence – C#

```
public class HandPresence : MonoBehaviour
{
    ...

    void UpdateHandAnimation()
    {
        if(targetDevice.TryGetFeatureValue(CommonUsages.trigger, out float triggerValue))
        {
            handAnimator.SetFloat("Trigger", triggerValue);
        }
        else
        {
            handAnimator.SetFloat("Trigger", 0);
        }

        if (targetDevice.TryGetFeatureValue(CommonUsages.grip, out float gripValue))
        {
            handAnimator.SetFloat("Grip", gripValue);
        }
        else
        {
            handAnimator.SetFloat("Grip", 0);
        }
    }

    ...
}
```

Programski kod 5.3: CloneHandController – C#

```
public class CloneHandController : MonoBehaviour
{
    ...

    private void Update()
    {
        float secondaryValue = secondaryAction.action.ReadValue<float>();
        if (secondaryValue != 0f)
        {
            cloneHand.GetComponent<Transform>().position = transform.position;
            cloneHand.GetComponent<Transform>().rotation = transform.rotation;
        }
    }
}
```

5.2 Ruke za zamrzavanje

Kako je navedeno, postoji više različitih kloniranih ruku. Jedne od njih su ruke za zamrzavanje, prikazane na slikama 5.2. Spomenute ruke igrač može stvoriti na poziciji svojih ruku pritiskom na Y/B gumb svojih kontrolera. One omogućavaju zamrzavanje određenih pokretnih objekata na mjestu, u svrhu rješavanja zagonetki.

a) Desna ruka *b) Lijeva ruka*

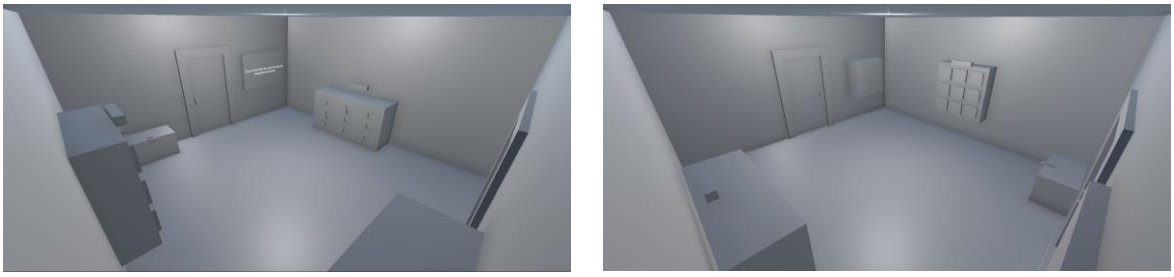
Slika 5.2: Ruke za zamrzavanje

Ruke za zamrzavanje imaju sferični *collider* koji može zamrznuti druge objekte na mjestu. Objekti koji se smiju zamrznuti su određeni skriptom zvanom CanBeFrozenController. Skripta CanBeFrozenController, prikazana na kodu 5.4, provjerava koji objekti uđu u *collider* i sprema ih u listu. Sadržaj se liste zatim provjerava. Ako lista sadrži ruku za zamrzavanje objekt se zamrzava na mjestu i ostaje zamrznut dok se klonirana ruka ne ukloni.

Programski kod 5.4: CanBeFrozenController – C#

```
public class CanBeFrozenController : MonoBehaviour
{
    private Rigidbody rb;
    private List<Collider> hitColliders = new List<Collider>();

    private void OnTriggerEnter(Collider otherObject)
    {
        if (!hitColliders.Contains(otherObject))
        {
            hitColliders.Add(otherObject);
        }
    }

    private void OnTriggerExit(Collider otherObject)
    {
        if (hitColliders.Contains(otherObject))
        {
            hitColliders.Remove(otherObject);
        }
    }

    ...

    void Update()
    {
        if (hitColliders.Find(item => item.gameObject.tag == "Freeze Hand") != null)
        {
            rb.isKinematic = true;
        } else rb.isKinematic = false;
    }
}
```

Primjer prostorije gdje se koriste ruke za zamrzavanje, prikazana na slikama 5.3, započinje s ladicama. U jednoj se od ladica nalazi ključ za otključavanje ormarića. U tome se ormariću nalazi ključ za otključavanje kovčega u kojem je lopta. Igrač shvaća da lopta nema detektor objekta iste boje kao lopta te ju donosi blizu lokota jednake boje, kako bi ga otključao. U toj ladici pronalazi ispravnu kombinaciju za kombinacijske gumbe, koje rješava zamrzavanjem dva gumba na mjestu i pritiskom na preostala dva. Gumbi nakon rješavanja otključavaju drugu ladicu u kojoj se nalazi ključ za posljednju ladicu. Nakon pokušavanja otključavanja posljednje ladice igrač pronalazi skriven detektor objekta u koji ubacuje ključ i otključava ladicu. U toj ladici pronalazi ispravnu kombinaciju za kombinacijske ladice koje također rješava zamrzavanjem dvije ladice i time napokon otključava vrata i nastavlja dalje.



a) *b)*

Slika 5.3: Primjer prostorije za ruke za zamrzavanje

5.3 Ruke za penjanje

Drugi tip kloniranih ruku su ruke za penjanje, prikazane na slikama 5.4. Spomenute ruke igrač može stvoriti na poziciji svojih ruku pritiskom na Y/B gumb svojih kontrolera. One omogućavaju igraču penjanje kada ljestve ili drugi objekti za penjanje nisu prisutni. Kako bi se penjao, igrač se samo treba primiti za klonirane ruke.

a) Desna ruka *b) Lijeva ruka*

Slika 5.4: Ruke za penjanje

Ruke za penjanje sadrže točku za koju se igrač može primiti kontrolerima kako bi se penjao. Skripte koje to omogućuju zovu se Climbable, Climber i ControllerVelocity. Climbable i ControllerVelocity skripte se nalaze na rukama za penjanje dok se Climber

nalazi na kontrolerima. Climber skripta, prikazana na kodu 5.5, omogućuje igraču penjanje tako da, ako se igrač primi za objekt koji sadrži skriptu Climbable, on se može odgurnuti od toga objekta u smjeru u kojem želi ići. Skripta dobiva uputstva kojom rukom se igrač pridržava za objekt od skripte Climbable. ControllerVelocity skripta, prikazana na kodu 5.6, služi samo da mjeri trenutnu brzinu kretanja kontrolera. Ta se vrijednost šalje u Climber skriptu i invertira kao negativna sila na igrača dok se penje. Climbable skripta, prikazana na kodu 5.7, stavlja se na objekte pomoću kojih se igrač smije penjati. Ona detektira skriptu Climber na igračevim rukama i određuje kojom rukom se trenutačno pridržava za objekt. Primjer izvođenja prikazan je na [14], od kuda su skripte prekopirane i modificirane za korištenu verziju Unity programa.

Programski kod 5.5: Climber – C#

```
public class Climber : MonoBehaviour
{
   …

   void FixedUpdate()
   {
      if(climbingHand)
      {
         continuousMoveProvider.enabled = false;
         Climb();
      } else
      {
         continuousMoveProvider.enabled = true;
      }
   }

   void Climb()
   {
      velocity = climbingHand.GetComponent<ControllerVelocity>().velocity;
      character.Move(transform.rotation * -velocity * Time.fixedDeltaTime);
   }
}
```

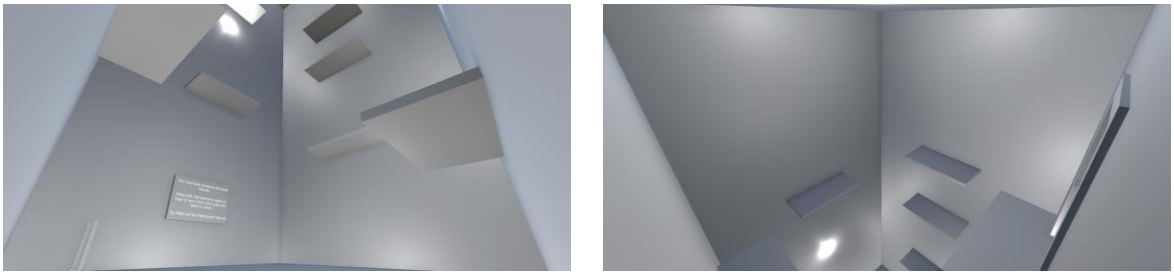
Programski kod 5.6: ControllerVelocity – C#

```
public class ControllerVelocity : MonoBehaviour
{
   public InputActionProperty velocityValue;
   public Vector3 velocity = Vector3.zero;

   void Update()
   {
      velocity = velocityValue.action.ReadValue<Vector3>();
   }
}
```

Programski kod 5.7: Climbable – C#

```
public class Climbable : XRBaseInteractable
{
   protected override void OnSelectEntered(SelectEnterEventArgs args)
   {
      base.OnSelectEntered(args);

      if(args.interactorObject is XRDirectInteractor)
      {
         Climber.climbingHand =
args.interactorObject.transform.gameObject.GetComponent<ActionBasedController>();
      }
   }

   protected override void OnSelectExited(SelectExitEventArgs args)
   {
      base.OnSelectExited(args);

      if(args.interactorObject is XRDirectInteractor)
      {
         if(Climber.climbingHand && Climber.climbingHand.name ==
args.interactorObject.transform.gameObject.name)
         {
            Climber.climbingHand = null;
         }
      }
   }
}
```

Primjer prostorije gdje igrač koristi ruke za penjanje, prikazana na slikama 5.5, započinje penjanjem uz pomoć ljestava. U prostoriji se uz zidove nalaze mnoge platforme koje igraču služe kao odmorišta tijekom penjanja. Igračev cilj je doći do vrha prostorije uz pomoć ruku za penjanje, prelazeći s platforme na platformu te na kraju prolazeći kroz otključana vrata.



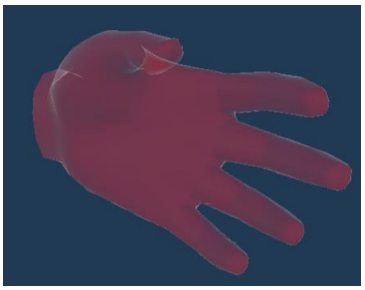
a) *b)*

Slika 5.5: Primjer prostorije za ruke za penjanje

5.4 Ruke za teleportaciju

Treći i zadnji tip kloniranih ruku su ruke za teleportaciju, prikazane na slikama 5.6. Spomenute ruke igrač može stvoriti na poziciji svojih ruku pritiskom na Y/B gumb svojih kontrolera. One omogućavaju igraču prebacivanje na njihovu poziciju po želji pritiskom na X/A gumb.

a) Desna ruka



b) Lijeva ruka

Slika 5.6: Ruke za teleportaciju

Ruke za teleportaciju sadrže *collider* koji sprječava igrača da se teleportira, ako ima objekata unutar *collidera* koji bi ga zarobili. Skripta koja upravlja *colliderom* zove se TeleportHandCollider, dok skripta koja omogućava teleportaciju i provjerava dozvoljenost teleportacije zove se TeleportHandController. Objekti koji se smiju teleportirati sadrže CanBeTeleported skriptu. TeleportHandCollider skripta, prikazana na kodu 5.8, upravlja samim *colliderom* tako da se pobrinjava kako god je ruke za teleportaciju okrenuta on je uvijek uspravan. TeleportHandController skripta, prikazana na kodu 5.9, upravlja samom igračevom teleportacijom. Ona provjerava ima li stranih objekata unutar *collidera*. Ako *collider* sadrži strane objekte igrač se ne može teleportirati i mora premijestiti ruku za teleportaciju. Međutim, ako nema nikakvih stranih objekata, igrač se teleportira na poziciju ruke za teleportaciju. CanBeTeleported skripta, prikazana na kodu 5.10, određuje koji objekti se smiju teleportirati zajedno s igračem. Svi objekti sa skriptom na sebi smiju se teleportirati. Skripta provjerava ako igrač drži objekt. Ako ga drži, skripta teleportira objekt zajedno s igračem tijekom teleportacije.

Programski kod 5.8: TeleportHandCollider – C#

```
public class TeleportHandCollider : MonoBehaviour
{
   void Update()
   {
      gameObject.transform.rotation = Quaternion.Euler(-transform.parent.rotation.x, -transform.parent.rotation.y, -
transform.parent.rotation.z);
   }
}
```

Programski kod 5.9: TeleportHandController – C#

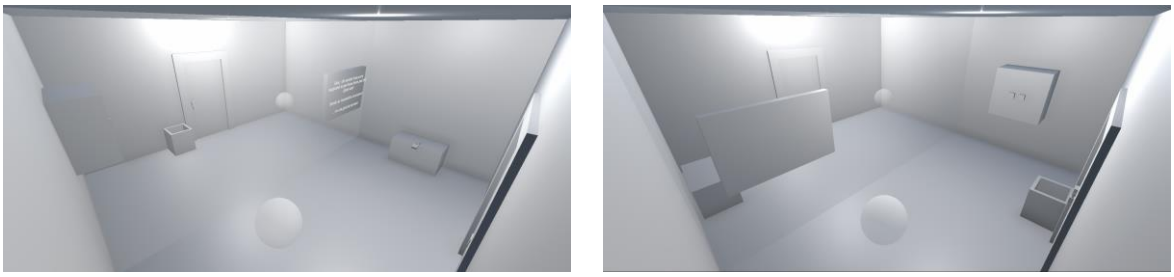
```
public class TeleportHandController : MonoBehaviour
{
   [SerializeField] private InputActionProperty secondaryAction;
   private List<Collider> hitColliders = new List<Collider>();
   public GameObject player, leftHand, rightHand;

   private void OnTriggerEnter(Collider otherObject)
   {
      if (!hitColliders.Contains(otherObject))
      {
```

```
            hitColliders.Add(otherObject);
        }
    }

    private void OnTriggerExit(Collider otherObject)
    {
        if (hitColliders.Contains(otherObject))
        {
            hitColliders.Remove(otherObject);
        }
    }

    void Update()
    {
        float secondaryValue = secondaryAction.action.ReadValue<float>();
        if (secondaryValue != 0f)
        {
            if(hitColliders.Count == 0)
            {
                player.transform.position = transform.parent.position + new Vector3(0, -1, 0);
                leftHand.transform.position = transform.parent.position + new Vector3(0, -0.5f, 0);
                rightHand.transform.position = transform.parent.position + new Vector3(0, -0.5f, 0);
            }
        }
    }
}
```

Programski kod 5.10: CanBeTeleported – C#

```
public class CanBeTeleported : MonoBehaviour
{
    [SerializeField] private InputActionProperty secondaryAction, secondaryAction2;
    public bool isGrabbed;
    private HandPresence hand;

    void Update()
    {
        float secondaryValue = secondaryAction.action.ReadValue<float>();
        float secondaryValue2 = secondaryAction2.action.ReadValue<float>();
        hand = FindObjectOfType<HandPresence>();
        if (secondaryValue != 0f || secondaryValue2 != 0f)
        {
            if (isGrabbed)
            {
                transform.position = hand.transform.position;
            }
        }
    }

    public void GrabObject()
    {
        isGrabbed = true;
    }

    public void DropObject()
    {
        isGrabbed = false;
    }
}
```

Primjer prostorije gdje igrač koristi ruke za teleportaciju, prikazana na slikama 5.7, provjerava igračevo razumijevanje teleportera za ruku i samih ruku za teleportiranje dok istovremeno pokazuje igraču da se objekti prenose s njim tijekom teleportacije. Igrač mora prijeći na drugu polovicu koristeći teleporter ruku i ruke za teleportaciju, kako bi dohvatio ključ za kovčeg na prvoj polovici u kojemu se nalazi lopta. Zatim uzima loptu i prenosi je na drugu polovicu, stavlja je u detektor objekta i nastavlja dalje.



a) *b)*

Slika 5.7: Primjer prostorije za ruke za teleportaciju

6. ZAKLJUČAK

Cilj ovoga rada bio je predstaviti drugačiju ideju mogućih mehanika tijekom igranja VR igre. Ideja je bila napraviti igru u kojoj igrač može stvoriti na poziciji svojih ruku kopije ruku. One mu daju nove mogućnosti, proširujući VR-ove već postojeće mogućnosti igračevih ruku, njihovo praćenje i korištenje.

Jedan od izazova tijekom izrade bila je verzija Unity programa na kojemu je rad započet. Bilo je potrebno promijeniti verziju programa sa 2021.3 na 2022.3 zbog novije verzije XR paketa i prilagoditi kod i funkcije novijoj verziji. Drugi izazov bio je izrada ruku za penjanje. Kod koji se nalazio unutar XR paketa nije omogućavao penjanje kakvo je bilo potrebno, to jest igrač je mogao nesmetano proći kroz zidove. Rješenje je bilo pronalaženje starije metode izrade penjanja i nadograđivanje metode za noviju verziju zbog zastarjelosti nekih funkcija.

Izvorni plan izrade igre bio je izraditi *escape room* igru u kojoj igrač ima četiri ruke. Mogao bi pritiskom na gumb mijenjati kontrolu, odnosno birati koju će lijevu ili koju će desnu ruku koristiti. Igrač bi mogao primiti objekt i zadržati ga s prvom lijevom rukom i prebaciti kontrolu na drugu lijevu ruku, radi drugog zadatka. Zbog izazova izrade takvih mehanika, odlučeno je stvoriti holografske kopije igračevih ruku koje bi funkcionirale slično. Prvo su osmišljene ruke za zamrzavanje, zatim su dodane ruke za penjanje i teleportaciju kako bi se koncept kloniranih ruku dodatno proširio. Same mogućnosti ruku mogu se dalje proširiti s više tipova ruku ili izradom novih različitih zagonetki tako da se različite ruke više koriste ili ograničavaju igračevo kretanje.

7. LITERATURA

[1] Wikipedia. Escape Room - Wikipedia [Online]. 2024. Dostupno na: https://en.wikipedia.org/wiki/Escape_room. (1.9.2024)

[2] Cluetivity. Escape Room Safety: How to Keep the Players Safe [Online]. 2024. Dostupno na: https://www.cluetivity.com/blog/escape-room-safety-how-to-keep-the-players-safe/. (1.9.2024)

[3] 99TH VR. The Room VR: A Dark Matter | Full Game Walkthrough | No Commentary [Online]. 2024. Dostupno na: https://www.youtube.com/watch?v=-20yThoegaE (12.9.2024)

[4] Unity. Unity – Manual: Unity User Manual [Online]. 2022. Dostupno na: https://docs.unity3d.com/Manual/index.html. (28.8.2024)

[5] Microsoft. Using Visual Studio Tools For Unity [Online]. 2021. Dostupno na: https://learn.microsoft.com/en-us/visualstudio/gamedev/unity/get-started/using-visual-studio-tools-for-unity?pivots=windows. (28.8.2024)

[6] Unity. Unity – Manual: XR [Online]. 2022. Dostupno na: https://docs.unity3d.com/Manual/XR.html. (28.8.2024)

[7] Git. Git – Reference [Online]. 2024. Dostupno na: https://git-scm.com/docs. (30.8.2024)

[8] Unity Discussion. Changing the color of all children in an empty gameobject [Online]. 2014. Dostupno na: https://discussions.unity.com/t/changing-the-color-of-all-children-in-an-empty-gameobject/108318. (28.8.2024)

[9] Fist Full of Shrimp. Unity VR Game Basics - PART 9 - Doors and Drawers [Online]. 2022. Dostupno na: https://youtu.be/jb63iw5tOec. (28.8.2024)

[10] RoBust Games. UNITY'S HINGE JOINT TUTORIAL 3D [Online]. 2021. Dostupno na: https://youtu.be/yIvJrcOOdD4. (28.8.2024)

[11] Sunny Valley Studio. Unity VR Stop Head Clipping with Fade-to-Black Effect [Online]. 2023. Dostupno na: https://youtu.be/YRjfmblMj8Q. (28.8.2024)

[12] Sunny Valley Studio. Unity VR Tutorial: Prevent Head Clipping by adding Push-Back effect [Online]. 2023. Dostupno na: https://youtu.be/FVPnp3fTGnw. (28.8.2024)

[13] Valem Tutorials. How to Make Physics Hands in VR - PART 1 - Unity VR Tutorial [Online]. 2022. Dostupno na: https://youtu.be/VG8hLKyTiJQ. (28.8.2024)

[14] Valem. Introduction to VR in Unity - PART 9 : CLIMBING [Online]. 2020. Dostupno na: https://youtu.be/mHHYI7hzZ6M. (28.8.2024)

8. OZNAKE I KRATICE

XR – Proširena Stvarnost (engl. Extended Reality)

VR – Virtualna Stvarnost (engl. Virtual Reality)

VUB – Veleučilište u Bjelovaru

RGB – Crvena-Zelena-Plava (engl. Red-Green-Blue)

9. SAŽETAK

Naslov:

Ova igra izvedena je u Unity programu za razvoj videoigara zajedno s Visual Studiom i Unity XR paketom.
Igra je izrađena u stilu *escape room* igre, gdje je igračev cilj tijekom igre riješiti zagonetke u svakoj prostoriji kako bi nastavio dalje.
Igra se sastoji od različitih elemenata kojima se igrač koristi kako bi riješio zagonetke i nastavio dalje kroz igru. Neki od najvažnijih elemenata su lokoti, ključevi i kombinacijski elementi kao ladice, gumbi i sat.
Klonirane ruke omogućavaju igraču kretanje i rješavanje zagonetki na nove načine, razlikujući se od drugih metoda tako da su klonirane ruke prenosive. Igrač samo treba pritisnuti gumb na upravljaču kako bi ih prenesao na drugu lokaciju.
Tri tipa kloniranih ruku su ostvareni. Prve imaju mogućnost zamrzavanja određenih objekata na mjestu ili u određenoj poziciji, druge penjanje bilo gdje je potrebno i treće teleportaciju igrača i prenošenje objekata kroz zidove. Uz svaki tip ruke također je opisana jedna prostorija u kojoj se te ruke koriste kao primjer njihove moguće upotrebe.

Ključne riječi: Unity program za razvoj videoigara, Visual Studio, XR paket, VR igra.

10. ABSTRACT

Title:

This game was developed in the Unity game development engine together with Visual Studio and the Unity XR package.

The game was made in the style of an escape room, where the player's goal throughout the game is to solve puzzles in each room to move forward.

The game consists of different elements that the player uses to solve puzzles and progress through the game. Some of the most important elements are padlocks, keys and combination elements such as drawers, buttons and a clock.

Cloned hands allow the player to move and solve puzzles in new ways, different from other methods in that cloned hands are portable. The player just needs to press a button on the controller to transport them to another location.

Three types of cloned hands have been realized. The first have the ability to freeze certain objects in place or in a certain position, the second to climb anywhere necessary, and the third to teleport the player and transfer objects through walls. With each type of hand one room where these hands are used is described as an example of their possible use.

Keywords: Unity game development engine, Visual Studio, XR package, VR game.

IZJAVA O AUTORSTVU ZAVRŠNOG RADA

Pod punom odgovornošću izjavljujem da sam ovaj rad izradio/la samostalno, poštujući načela akademske čestitosti, pravila struke te pravila i norme standardnog hrvatskog jezika. Rad je moje autorsko djelo i svi su preuzeti citati i parafraze u njemu primjereno označeni.

Mjesto i datum	Ime i prezime studenta/ice	Potpis studenta/ice
U Bjelovaru, 16. rujna 2024.	Leo Ružić	Leo Ružić

U skladu s čl. 58, st. 5 Zakona o visokom obrazovanju i znanstvenoj djelatnosti, Veleučilište u Bjelovaru dužno je u roku od 30 dana od dana obrane završnog rada objaviti elektroničke inačice završnih radova studenata Veleučilišta u Bjelovaru u nacionalnom repozitoriju.

Suglasnost za pravo pristupa elektroničkoj inačici završnog rada u nacionalnom repozitoriju

Leo Rožić
ime i prezime studenta/ice

Dajem suglasnost da tekst mojeg završnog rada u repozitorij Nacionalne i sveučilišne knjižnice u Zagrebu bude pohranjen s pravom pristupa (zaokružiti jedno od ponuđenog):

a) Rad javno dostupan
b) Rad javno dostupan nakon _____ (upisati datum)
c) Rad dostupan svim korisnicima iz sustava znanosti i visokog obrazovanja RH
d) Rad dostupan samo korisnicima matične ustanove (Veleučilište u Bjelovaru)
e) Rad nije dostupan

Svojim potpisom potvrđujem istovjetnost tiskane i elektroničke inačice završnog rada.

U Bjelovaru, 16. rujna 2024.

Leo Rožić
potpis studenta/ice